VFORK

Vulnerable to race conditions. Don't use vfork() in your programs.

Sean Barnum, Cigital, Inc. [vita[1]]

2007-04-23

Part "Original Cigital Coding Rule in XML"

Mime-type: text/xml, size: 5185 bytes

Attack Category	• Denial of Service
Vulnerability Category	• Race Condition • Process management • Unconditional
Software Context	• Process Management
Location	• unistd.h
Description	From http://www.linuxsecurity.com/HOWTO/Secure-Programs-HOWTO/avoid-vfork.html: The portable way to create new processes in Unix-like systems is to use the fork(2) call. BSD introduced a variant called vfork(2) as an optimization technique. In vfork(2), unlike fork(2), the child borrows the parent's memory and thread of control until a call to execve(2V) or an exit occurs; the parent process is suspended while the child is using its resources. The rationale is that in old BSD systems, fork(2) would actually cause memory to be copied while vfork(2) would not. Linux never had this problem; because Linux used copy-on-write semantics internally, Linux only copies pages when they changed (actually, there are still some tables that have to be copied; in most circumstances their overhead is not significant). Nevertheless, since some programs depend on vfork(2), recently Linux implemented the BSD vfork(2) semantics (previously vfork(2) had been an alias for fork(2)). There are a number of problems with vfork(2). From a portability point of view, the problem with vfork(2) is that it's actually fairly tricky for a process to not interfere with its parent, especially in high-level languages. The "not interfering" requirement applies to the actual machine code generated, and many compilers generate hidden temporaries and other code structures that cause unintended interference. The result: programs using vfork(2) can easily fail

1. http://buildsecurityin.us-cert.gov/bsi-rules/35-BSI.html (Barnum, Sean)

	when the code changes or even when compiler versions change. For secure programs it gets worse on Linux systems, because Linux (at least versions 2.2 through 2.2.17) is vulnerable to a race condition in vfork()'s implementation. If a privileged process uses a vfork(2)/execve(2) pair in Linux to execute user commands, there's a race condition while the child process is already running as the user's UID but hasn't entered execve(2) yet. The user may be able to send signals, including SIGSTOP, to this process. Due to the semantics of vfork(2), the privileged parent process would then be blocked as well. As a result, an unprivileged process could cause the privileged process to halt, resulting in a denial of service of the privileged process' service. FreeBSD and OpenBSD, at least, have code to specifically deal with this case, so they may not be vulnerable to this problem (Solar Designer noted and documented this problem in Linux on the ``security-audit'' mailing list on October 7, 2000). Don't use vfork(2) in your programs. It is or will be deprecated on some systems and is not portable on those systems that do support it.
APIs	**Function Name** / **Comments** vfork() / check fork() /
Method of Attack	Refer to Description.
Exception Criteria	
Solutions	**Solution Applicability** / **Solution Description** / **Solution Efficacy** Always / Use fork() instead of vfork() / Effective
Signature Details	pid_t vfork(void);
Examples of Incorrect Code	
Examples of Corrected Code	
Source Reference	• VFORK[3] (1999).
Recommended Resource	
Discriminant Set	**Operating System**: • UNIX **Languages**: • C • C++

Cigital, Inc. Copyright

The Build Security In (BSI) portal is sponsored by the U.S. Department of Homeland Security (DHS), National Cyber Security Division. The Software Engineering Institute (SEI) develops and operates BSI. DHS funding supports the publishing of all site content.

1. mailto:copyright@cigital.com